Detecting Structural Metadata with Decision Trees and Transformation-Based Learning

Joungbum Kim† and Sarah E. Schwarm‡ and Mari Ostendorf†
†Dept. of Electrical Engineering ‡Dept. of Computer Science
University of Washington
Seattle, WA 98195. USA
{bummie,sarahs,mo}@ssli.ee.washington.edu

Abstract

The regular occurrence of disfluencies is a distinguishing characteristic of spontaneous speech. Detecting and removing such disfluencies can substantially improve the usefulness of spontaneous speech transcripts. This paper presents a system that detects various types of disfluencies and other structural information with cues obtained from lexical and prosodic information sources. Specifically, combinations of decision trees and language models are used to predict sentence ends and interruption points and, given these events, transformation-based learning is used to detect edit disfluencies and conversational fillers. Results are reported on human and automatic transcripts of conversational telephone speech.

1 Introduction

Automatic speech-to-text (STT) transcripts of spontaneous speech are often difficult to comprehend even without the challenges arising from word recognition errors introduced by imperfect STT systems (Jones et al., 2003). Such transcripts lack punctuation that indicates clausal or sentential boundaries, and they contain a number of disfluencies that would not normally occur in written language. Repeated words, hesitations such as “um” and “uh”, and corrections to a sentence in mid-stream are a normal part of conversational speech. These disfluencies are handled easily by human listeners (Shriberg, 1994), but their existence makes transcripts of spontaneous speech ill-suited for most natural language processing (NLP) systems developed for text, such as parsers or information extraction systems. Similarly, the lack of meaningful segmentation in automatically generated speech transcripts makes them problematic to use in NLP systems, most of which are designed to work at the sentence level. Detecting and removing disfluencies and locating sentential unit boundaries in spontaneous speech transcripts can improve their readability and make them more suitable for NLP. Automatically annotating discourse markers and other conversational fillers is also likely to be useful, since proper handling is needed to follow the flow of conversation. Hence, the overall goal of our work is to detect such structural information in conversational speech using features generated by currently available speech processing systems and statistical machine learning tools.

This paper is organized as follows. In Section 2, we describe the types of metadata that this work addresses, followed by a discussion of related prior work in Section 3. Section 4 describes the system architecture and details the algorithms and features used by our system. Section 5 discusses the experimental paradigm and results. Finally we provide a summary and directions for future work in Section 6.

2 Structural Metadata

We consider three main types of structural metadata: sentence-like units, conversational fillers and edit disfluencies. These structures were chosen primarily because of the availability of annotated conversational speech data from the Linguistic Data Consortium (Strassel, 2003) and standard scoring tools (NIST, 2003).

2.1 Sentence Units

Conversational speech lacks the clear sentence boundaries of written text. Instead, we detect *SUs* (variously referred to as sentence, semantic, and slash units), which are linguistic units maximally equivalent to sentences that are used to mark segmentation boundaries in conversational speech where utterances often end without forming “grammatical” sentences in the sense expected in written text. SUs can be sub-categorized according to their discourse role. In our data, annotations distinguish statement, question, backchannel, incomplete SU and SU-internal clause boundaries. Here, we ignore the SU-internal boundaries, and merge all but the incomplete SU categories in characterizing SU events.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **2004**	2. REPORT TYPE	3. DATES COVERED **00-00-2004 to 00-00-2004**

4. TITLE AND SUBTITLE **Detecting Structural Metadata with Decision Trees and Transformation-Based Learning**	5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **University of Washington,Computer Science and Engineering,Box 352350,Seattle,WA,98195-2350**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**8**	

Table 1: Filled pauses and discourse markers to be detected by our system.

Filled Pauses	ah, eh, er, uh, um
Discourse Markers	actually, anyway, basically, I mean, let's see, like, now, see, so, well, you know, you see

Table 2: Examples of edit disfluencies.

Disfluency	**Example**
Repetition	(I was) + **I was** very interested... (I was) + { uh} **I was** very interested...
Repair	(I was) + **she was** very interested... (I was) + { I mean } **she was** very...
Restart	(I was very) + Did you hear the news?

2.2 Conversational Fillers

Conversational fillers include filled pauses (hesitation sounds such as "uh", "um" and "er"), discourse markers (e.g. "well", "you know"), and explicit editing terms. Defining an all-inclusive set of English filled pauses and discourse markers is a problematic task. Our system detects only a limited set of filled pauses and discourse markers, listed in Table 1, which cover a large majority of cases (Strassel, 2003). An explicit editing term is a filler occurring within an edit disfluency, described further below. For example, the discourse marker *I mean* serves as an explicit editing term in the following edit disfluency: "I didn't tell her that, *I mean*, I couldn't tell her that he was already gone."

2.3 Edit Disfluencies

Edit disfluencies largely encompass three separate phenomena: repetition, repair and restart (Shriberg, 1994). A *repetition* occurs when a speaker repeats the most recently spoken portion of an utterance to hold off the flow of speech. A *repair* happens when the speaker attempts to correct a mistake that he or she just made. Finally, in a *restart*, the speaker abandons a current utterance completely and starts a new one.

Previous studies characterize edit disfluencies using a structure with different segments (Shriberg, 1994; Nakatani and Hirschberg, 1994). The first part of this structure is called the *reparandum*, a string of words that gets repeated or corrected. The reparandum is immediately followed by a non-lexical boundary event termed the *interruption point* (IP). The IP marks the point where the speaker interrupts a fluent utterance. Optionally, there may be a filled pause or explicit editing term. The final part of the edit disfluency structure is called the *alteration*, which is a repetition or revised copy of the reparandum. In the case of a restart, the alteration is empty. In Table 2, reparanda are enclosed in parentheses, IPs are represented by "+", optional fillers are in braces, and alterations are in boldface.

Annotation of complex edit disfluencies, where a disfluency occurs within an alteration, can be difficult. The data used here is annotated with a flattened structure that treats these cases as simple disfluencies with multiple IPs (Strassel, 2003). IPs within a complex disfluency are detected separately, and contiguous sequences of edit words associated with these IPs are referred to as a deletable region.

3 Previous Work

In an early study on automatic disfluency detection a deterministic parser and correction rules were used to clean up edit disfluencies (Hindle, 1983). However theirs was not a truly automatic system as it relied on hand-annotated "edit signals" to locate IPs. Bear et al. (1992) explored pattern matching, parsing and acoustic cues and concluded that multiple sources of information would be needed to detect edit disfluencies. A decision-tree-based system that took advantage of various acoustic and lexical features to detect IPs was developed in (Nakatani and Hirschberg, 1994).

Shriberg et al. (1997) applied machine prediction of IPs with decision trees to the broader Switchboard corpus by generating decision trees with a variety of prosodic features. Stolcke et al. (1998) then expanded the prosodic tree model with a hidden event language model (LM) to identify sentence boundaries, filled pauses and IPs in different types of edit disfluencies. The hidden event LM used in their work adapted Hidden Markov Model (HMM) algorithms to an n-gram LM paradigm to represent non-lexical events such as IPs and sentence boundaries as hidden states. Liu et al. (2003) built on this framework and extended prosodic features and the hidden event LM to predict edit IPs on both human transcripts and STT system output. Their system also detected the onset of the reparandum by employing rule-based pattern matching once edit IPs have been detected.

Edit disfluency detection systems that rely exclusively on word-based information have been presented by Heeman et al. (Heeman et al., 1996) and Charniak and Johnson (Charniak and Johnson, 2001). Common to both of these approaches is a focus on repeated or similar sequences of words and information about the words themselves and the length and similarity of the sequences.

Our approach is most similar to (Liu et al., 2003), since we also detect boundary events such as IPs first and use them as "signals" when identifying the reparandum in a later stage. The motivation to detect IPs first is that

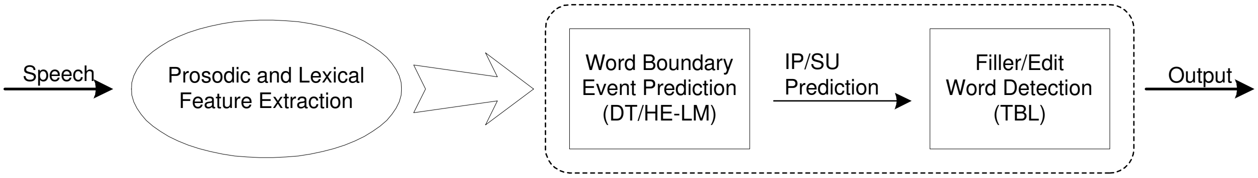
Speech
Prosodic and Lexical Feature Extraction
Word Boundary Event Prediction (DT/HE-LM)
IP/SU Prediction
Filler/Edit Word Detection (TBL)
Output

Figure 1: System Diagram

speech before an IP is fluent and is likely to be free of any prosodic or lexical irregularities that can indicate the occurrence of an edit disfluency. Like Liu et al., we use a decision tree trained with prosodic features and a hidden event language model for the IP detection task. However, we incorporate SU detection in those models as well. We use part-of-speech (POS) tags and pattern match features in decision tree training whereas Liu et al. (2003) developed language models for them. We explore three different methods of combining the hidden event language model and the decision tree model, namely linear interpolation, joint tree-based modeling and an HMM-based approach. Moreover, our system uses the transformation-based learning algorithm rather than hand-crafted rules for the second stage of edit region detection.

Another key difference between our system and most previous work is the prediction target. Our system incorporates detecting word boundary events such as SUs and IPs, locating onsets of edit regions, and identifying filled pauses, discourse markers and explicit editing terms. We believe that such a comprehensive detection scheme allows our system to better model dependencies between these events, which will lead to an improvement in the overall detection performance.

4 System Description

4.1 Overall Architecture

As shown in Figure 1, our system detects disfluencies in a two-step process. First, for each word boundary in the given transcription, a decision tree predicts one of the four boundary events IP, SU, ISU (incomplete SU), and the null event. Then in the second stage, rules learned via the transformation-based learning (TBL) algorithm are applied to the data containing predicted boundary events and other lexical information to identify edits and fillers. Following edit region and filler prediction, the system output was post-processed to eliminate edit region predictions not associated with IP predictions as well as IP predictions for which no edit region or filler was detected. An analysis of post-processing alternatives confirmed that this strategy reduced insertion errors.

4.2 Detecting Boundary Events

In order to detect boundary events, we trained a CART-style decision tree (Breiman et al., 1984) with various prosodic and lexical features. Decision trees are well-suited for this task because they provide a convenient way to integrate both symbolic and numerical features in prediction. Furthermore, a trained decision tree is highly explainable by its nature, which allows us to gain additional insight into the utilities of and the interactions between multiple information sources.

Prosodic features generated for decision tree training included the following:

- Word and rhyme[1] durations.
- Rhyme duration differences between two neighboring words.
- F0 statistics (minimum, mean, maximum, slope) over a word.
- Differences in F0 statistics between two neighboring words.
- Energy statistics over a word and its rhyme.
- Silence duration following a word.
- A flag indicating start and end of a speaker turn and speaker overlap.
- Ordinal position of a word in a turn.

Energy and F0 features were generated with the Entropic System ESPS/Waves package and the F0 stylization tool developed in (Sönmez et al., 1998). Word and rhyme duration were normalized by phone duration statistics (mean and variance) calculated over all available training data. F0 and energy features were normalized for each individual speaker's baseline. A turn boundary was hypothesized for word boundaries with silences longer than four seconds.

Since inclusion of features that do not contribute to the classification of data can degrade the performance of a decision tree, we selected only the prosodic features whose exclusion from the training process led to a decrease in boundary event detection accuracy on the development data by utilizing the leave-one-out method.

Lexical features consisted of POS tag groups, word and POS tag pattern matches, and a flag indicating existence

[1] In our work, a rhyme was defined to contain the final vowel of a word and any consonants following the final vowel.

of filler words to the right of the current word boundary. The POS tag features were produced by first predicting the tags with Ratnaparkhi's Maximum Entropy Tagger (Ratnaparkhi, 1996) and then clustered by hand into a smaller number of groups based on their syntactic role. The clustering was performed to speed up decision tree training as well as to reduce the impact of tagger errors.

Word pattern match features were generated by comparing words over the range of up to four words across the word boundary in consideration. Grouped POS tags were compared in a similar way, but the range was limited to at most two tags across the boundary since a wider comparison range would have resulted in far more matches than would be useful due to the low number of available POS tag groups. When words known to be identified frequently as fillers existed after the boundary, they were skipped and the range of pattern matching was extended accordingly.

Another useful cue for boundary event detection is the existence of word fragments. Since word fragments occur when the speaker cuts short the word being spoken, they are highly indicative of IPs. However currently available STT systems do not recognize word fragments. As our goal is to build an automatic detection system, our system was not designed to use any features related to word fragments. However, for a control case, we conducted an experiment with reference transcripts using a single "frag" word token to show the potential for improved performance of a system capable of recognizing fragments.

In addition to the decision tree model, we also employed a hidden event language model to predict boundary events. A hidden event LM is the same as a typical n-gram LM except that it models non-lexical events in the n-gram context by counting special non-word tokens representing such events. The hidden event LM estimates the joint distribution $P(W, E)$ of words W and events E. Once the model has been trained, a forward-backward algorithm can be used to calculate $P(E|W)$, or the posterior probability of an event given the preceding word sequence (Stolcke et al., 1998; Stolcke and Shriberg, 1996). The SRI Language Modeling Toolkit (SRILM) (Stolcke, 2002) was used to train a trigram open-vocabulary language model with Kneser-Ney discounting (Kneser and Ney, 1995) on data that had boundary events (SU, ISU, and IP) inserted in the word stream. Posterior probabilities of boundary events for every word boundary were then estimated with SRILM's capability for computing hidden event posteriors.

While the hidden event LM alone can be used to detect boundary events, prior work has shown that it benefits from also using prosodic cues, so we combined the language model and the decision tree model in three different ways. In the first approach, which we call the joint tree model, the boundary event posterior probability from the hidden event LM is jointly modeled with other features in the decision tree to make predictions about the boundary events. In the second approach, referred to as the linearly interpolated model, a decision is made based on the combined posterior probability

$$\lambda P_{tree}(E|A, W) + (1 - \lambda) P_{LM}(E|W),$$

where A corresponds to the acoustic-prosodic features and the weighting factor λ can be chosen empirically to maximize target performance, i.e. bias the prediction toward the more accurate model. In the third approach, the decision tree features, words and boundary events are jointly modeled via an integrated HMM (Shriberg et al., 2000). This approach augments the hidden event LM by modeling decision tree features as emissions from the HMM states represented by the word and boundary event. Under this framework, the forward-backward algorithm can again be used to determine posterior probabilities of boundary events. Similar to the linearly interpolated model, a weighting factor can be used to introduce the desired bias to the combination model. The joint tree model has the advantage that the (possibly) complex interaction between lexical and prosodic cues can be captured. However, since the tree is trained on reference transcriptions, it favors lexical cues, which are less reliable in STT output. In the linearly interpolated and joint HMM approaches, the relative weighting of the two knowledge sources is estimated on the development test set for STT output, so it is possible for prosodic cues to be given a higher weight.

4.3 Edit and Filler Detection

After SUs and IPs have been marked, we use transformation-based learning (TBL) to learn rules to detect edit disfluencies and conversational fillers. TBL is an automatic rule learning technique that has been successfully applied to a variety of problems in natural language processing, including part-of-speech tagging (Brill, 1995), spelling correction (Mangu and Brill, 1997), error correction in automatic speech recognition (Mangu and Padmanabhan, 2001), and named entity detection (Kim and Woodland, 2000). We selected TBL for our tagging-like metadata detection task since it has been used successfully for these other tagging tasks.

TBL is an iterative technique for inducing rules from training data. A TBL system consists of a baseline predictor, a set of rule templates, and an objective function for scoring potential rules. After tagging the training data using the baseline predictor, the system learns a list of rules to correct errors in these predictions. At each iteration, the system uses the rule templates to generate all possible rules that correct at least one error in the training data and selects the best rule according to the objective function, commonly token error rate. The best rule is

Table 3: Example word and POS matches for TBL.

Word Match	that IP that
POS Match	the dog IP the cat

recorded and applied to the training data in preparation for the next iteration. The standard stopping criterion for rule learning is to stop when the score of the best rule falls below a threshold value; statistical significance measures have also been used (Mangu and Padmanabhan, 2001). To tag new data, the rules are applied in the order in which they were learned. This allows rules which are learned later in the process to fine tune the effects of the earlier rules. TBL produces concise, comprehensible rules, and uses the entire corpus to train all of the rules. We used Florian and Ngai's Fast TBL system (fnTBL) (Ngai and Florian, 2001) to train rules using disfluency annotated conversational speech data.

The input to our TBL system consists of text divided into utterances, with IPs and SUs inserted as if they were extra words. (For simplicity, these special words are also assigned "IP" and "SU" as part of speech tags.)

Our TBL system used the following types of features:

- Identity of the word.
- Part of speech (POS) and grouped part of speech (GPOS) of the word (same as the decision tree).
- Is the word commonly used as: filled pause (FP), backchannel (BC), explicit editing term (EET), discourse marker (DM)?
- Does this word/ POS/ GPOS match the word/ POS/ GPOS that is 1/2/3 positions to its right?
- Is this word at the beginning of a turn or utterance?
- Tag to be learned.

The "tag" feature is the one we want the system to learn. It is also used in templates that consider features of neighboring words. The baseline predictor sets the tag to its most common value, "no disfluency," for all words. Other values of the tag are the three types of fillers (FP, EET, DM) and edit. The objective function for our learner is token error rate, and rule learning is stopped at a threshold score of 5.

We generated a set of rule templates using these features. The rule templates account for individual features of the current word and/or its neighbors, the proximity of potential FP/EET/DM terms, and matches between the current word and nearby words, especially when in close proximity to a boundary event or potential filler. Example word and POS matches are shown in Table 3.

5 Experiments

5.1 Experimental Setup

For training our system and its components, we used two different subsets of Switchboard, a corpus of conversational telephone speech (CTS) (Godfrey et al., 1992). One of the data sets included 417 conversations (LDC1.3) that were hand-annotated by the Linguistic Data Consortium for disfluencies and SUs according to the V5 guidelines detailed in (Strassel, 2003). Another set of 1086 conversations from the Switchboard corpus was annotated according to (Meteer et al., 1995) and is available as part of the Treebank3 corpus (TB3). We used a version of this set that contained annotations machine-mapped to approximate the V5 annotation specification.

For development and testing of our system, we used hand transcripts and STT system output for 72 conversations from Switchboard and the Fisher corpus, a recent CTS data collection. Half of these conversations were held out and used as development data (dev set), and the other 36 conversations were used as test data (eval set). The STT output, used only in testing, was from a state-of-the-art large vocabulary conversational speech recognizer developed by BBN. The word error rates for the STT output were 27% on the dev set and 25% on the eval set.

To assess the performance of our overall system, disfluencies and boundary events were predicted and then evaluated by the scoring tools developed for the NIST Rich Transcript evaluation task.

5.2 Boundary Event Prediction

Decision trees to predict boundary events were trained and tested using the IND system developed by NASA (Buntine and Caruan, 1991). All decision trees were pruned by ten-fold cross validation. The LDC1.3 set[2] with reference transcriptions was used to train the trees[3] and the dev set was used to evaluate their performances.

Several decision trees with different combinations of feature groups were trained to assess the usefulness of different knowledge sources for boundary event detection. The tree was then used to predict the boundary events on the reference transcription of the dev set. The results are presented in Table 4. The inclusion of a special token for fragments resulted in improved precision and recall for SUs and IPs but, surprisingly, degraded performance for ISUs. These results show that prosodic features by themselves failed to detect ISUs and IPs, though

[2] Experiments combining the LDC1.3 set with the mapped TB3 set were not as successful as LDC1.3 set alone for decision tree training.

[3] While it might be better to train from automatic transcripts, it is difficult to define target class labels in cases where there are insertion errors or a sequence of several word errors.

Table 4: Impact of different features on boundary event prediction using the joint tree model on reference transcripts.

Features	SU		ISU		IP	
	Recall	Precision	Recall	Precision	Recall	Precision
Prosody Only	46.5	74.6	0	-	8.8	47.2
POS, Pattern, LM	77.3	79.6	30.0	53.3	64.4	77.4
Prosody, POS, Pattern, LM	81.5	80.4	36.5	69.7	66.1	78.7
All Above + Fragments	81.1	81.6	20.1	60.7	80.7	80.4

they lead to performance gains when combined with lexical cues. Examination of the decision tree trained with only the prosodic features revealed that pause duration and turn information features were placed near the top of the tree.

Use of lexical features brought substantial performance improvement in all aspects, and classification accuracy increased when features extracted from different knowledge sources were combined. However, we observed that a smaller number of prosodic features ended up being used in the tree and they were placed at or near leaf nodes as more lexical features were made available for training. The importance of prosodic features is likely to be much more apparent for STT data. The word errors prevalent in the STT transcriptions will affect lexical features far more severely than prosodic features, and therefore the prosodic features contribute to the robustness of the overall system when lexical features become less reliable.

5.3 Edit and Filler Detection

After the prediction of boundary events, the rules learned by the TBL system described in section 4.3 were applied to detect fillers and edit regions. As with the decision trees, we trained rules using the LDC1.3 data alone, and combined with the mapped TB3 data, finding that the combined dataset gave better results for TBL training. Again we used only reference word transcripts but discovered that training with SUs and IPs predicted by the first stage of our system was more effective than using reference boundary events.

It is difficult to formally assess the effectiveness of the TBL module independently, and results for the entire system are discussed in detail in the next section. Informal inspection of the rules learned by the TBL system indicates that, not surprisingly, word match features and the presence of IPs are very important for the detection of edit regions. The most commonly used features for identifying discourse markers are the identity or POS of the current and/or neighboring words and the tag already assigned to neighboring words.

Table 5: Detection of boundary events and disfluencies on STT output as scored by *rt-eval*.

Task	% Corr	% Del	% Ins	% SER
Filler	63.9	36.1	14.0	50.1
Edit	25.5	74.5	13.7	88.2
IP	49.6	50.5	16.3	66.8
SU	73.1	26.9	19.7	46.6

5.4 Overall System Results

The performance of our system was evaluated on the fall 2003 NIST Rich Transcription Evaluation test set (RT-03F) using the *rt-eval* scoring tool (NIST, 2003), which combines ISUs and SUs in a single category, and reports results for detection of SUs, IPs, fillers, and edits without differentiating subcategories of fillers and edits. This tool produces a collection of results, including percentage correct, deletions, insertions, and *Slot Error Rate (SER)*, similar to the word error rate measure used in speech recognition. SER is defined as the number of insertions and deletions divided by the number of reference items. Note that scores are somewhat different from those in Table 4, because of differences in scoring and metadata alignment methods.

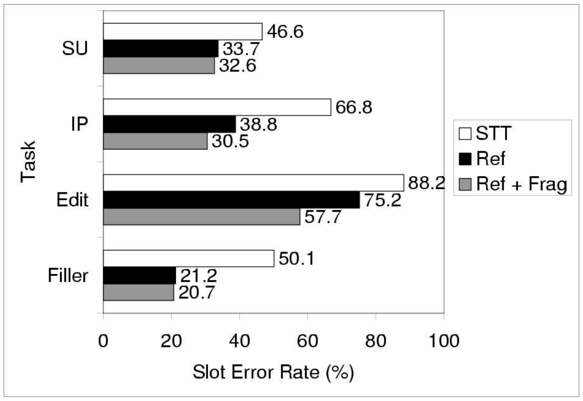

Figure 2: Detection of boundary events and disfluencies on reference and STT transcripts (joint tree model).

Results of our system on the RT-03F task are shown in

Table 6: Percentage of missed IPs on the dev set.

Transcription	% IPs after fragments	% Other edit IPs
Reference	81.7	37.6
STT	74.0	51.2

Table 5 for the joint tree version of the system as applied to the STT transcription of the test data. SU detection by our system is relatively good. IP detection is not as successful, which also impacts edit detection.

Figure 2 contrasts the results of the joint tree model for STT output with those obtained on reference data with and without fragments. As expected, all error rates are higher on STT output; IPs and fillers take the biggest hit. Filler performance in particular seems to be affected by recognition errors, which is not surprising, since misrecognized words would likely not be on the target lists of filled pauses and discourse markers. In particular, nearly all missed and incorrectly inserted filled pauses are due to recognition errors. Detection of discourse markers is more challenging; fewer than half the errors on discourse markers are due to recognition errors. Most non-STT-related filler errors involved the words "so" and "like" used as DMs, which are hard problems since the vast majority of the occurrences of these two words are not DMs. It is also not surprising that improved IP detection on reference data contributes to a lower error rate for edits.

As expected, the inclusion of fragments improves performance on IP and edit detection, where fragments frequently occur. In LDC1.3, 17.2% of edit IPs have word fragments occurring before them; 9.9% of edits consist of just a single fragment. In the dev set, 35.5% of edit IPs are associated with fragments. However, fragments are rarely output by the STT system, so for most of our work we chose to use the identical system for processing reference and STT transcripts and did not include fragments. IP detection performance was significantly worse for those IPs associated with fragments, as shown in Table 6. However, since fragments are often deleted or recognized as a full word, STT output actually "helps" with detection of IPs after fragments, apparently because the POS tagger and hidden event LM tend to give unreliable results on the reference transcripts near fragments.

Figure 3 compares the eval test set performances of the different alternatives for incorporating the hidden event LM posterior, i.e. inclusion in the decision tree, linear interpolation and the joint HMM. For this experiment, the interpolation weighting factor was selected empirically to maximize boundary event prediction accuracy on the STT transcription of the dev set. The results of this comparison are mixed: SU detection is better with the joint tree model, but IP detection and consequently edit detection are better with the interpolation and HMM approaches. The degradation of SU detection performance with the HMM is counter to findings in previous work (Stolcke et al., 1998; Shriberg et al., 2000). This may be due to differences in evaluation criteria, given that the HMM approach typically had higher precision which might benefit earlier word-based measures more. In addition, the difference in conclusions may be due to the fact that the decision trees used here include lexical pattern match features in addition to hidden event posteriors.

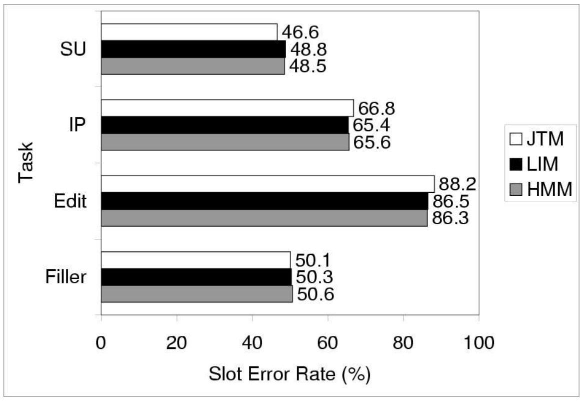

Figure 3: Results for joint tree (JTM), linearly interpolated (LIM) and joint HMM models on STT transcripts.

A problem in our system is the inability to predict more than one label for a given word or boundary. Words labeled as both filler and edit account for only 0.5% of all fillers and edits in the LDC1.3 training data, so it is probably not a significant problem. We also do not predict boundaries as both SU and IP. In LDC1.3, these account for 12.8% of SU boundaries, and are treated as simply SU in training. This does not affect IPs for edits, but impacts 38.6% of IPs before fillers. By predicting a combined SU-IP boundary in addition to isolated SUs and IPs, we obtain a small reduction in SER for IPs but at the expense of an increase in SU SER. However, separating prediction of IPs after edit regions vs. before fillers also yields small improvements in edit region precision and filler recall, resulting in 3.3% and 0.8% relative reduction in filler and edit SERs respectively for the joint HMM.

6 Conclusions

We have demonstrated a two-tiered system that detects various types of disfluencies in spontaneous speech. In the first tier, a decision tree model utilizes multiple knowledge sources to predict interword boundary events. Then the system employs a transformation-based learning algorithm to identify the extent and type of disfluencies. Experimental results show that the large variance and noise inherent in prosodic features makes them much less effective than lexical features for reference data; however, in the presence of word recognition errors prevalent in automatic transcripts of spontaneous speech, prosodic features have more value. Performance differences for the various score combination methods were small, but combining decision tree and HE-LM scores with a weight optimized on dev data is slightly better for edit disfluencies. Transformation-based learning is an effective way to tag fillers and edit regions after boundary events are tagged, but the best performance is obtained when training with automatically predicted SU and IP boundary events.

As this is a new task, error rates are relatively high (though significantly better than chance), but this approach achieved competitive results on the Fall 2003 NIST Rich Transcription Evaluation, and there are many directions for future improvements.

Acknowledgments

This work was supported by DARPA, no. MDA904-02-C-0437, in a project led by BBN. The authors thank their colleagues at BBN for providing recognizer output for the training and test data, and colleagues at SRI for providing F0 conditioning tools and mapped TB3 data. Any opinions, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsor or our collaborators.

References

J. Bear et al., ”Integrating multiple knowledge sources for detection and correction of repairs in human-computer dialog”, *Meeting of the ACL*, pp. 56–63, 1992.

L. Breiman et al., *Classification and Regression Trees* Chapman and Hall, 1984.

E. Brill, ”Transformation-based error-driven learning and natural language: a case study in part of speech tagging”, *Computational Linguistics*, 21(4), pp. 543–565, 1995.

W. Buntine and R. Caruan ”Introduction to IND and recursive partitioning”, NASA Ames Research Center, TR. FIA-91-28, 1991.

E. Charniak and M. Johnson, ”Edit detection and parsing for transcribed speech”, *Proc. NAACL*, pp. 118–126, 2001.

J. J. Godfrey et al., ”SWITCHBOARD: Telephone speech corpus for research and development”, *Proc. ICASSP*, v. I, pp. 517–520, 1992.

P. A. Heeman et al., ”Combining the detection and correction of speech repairs”, *Proc. ICSLP*, v. 1, pp. 362–365, 1996.

C. Hemphill et al., “The ATIS spoken language systems pilot corpus”, *Proc. of DARPA Speech and Natural Language Workshop*, pp. 96–101, 1990.

D. Hindle ”Deterministic parsing of syntactic nonfluencies”, *Meeting of the ACL*, pp. 123–128, 1983.

D. Jones et al., ”Measuring the readability of automatic speech-to-text transcripts”, *Proc. Eurospeech*, pp. 1585–1588, 2003.

J.-H. Kim and P. Woodland ”A rule-based named entity recognition system for speech input”, *Proc. ICSLP*, pp.2757–2760, 2001.

R. Kneser and H. Ney ”Improved backing-off for mgram language modeling”, *Proc. ICASSP*, pp. 181–184, 1995.

Y. Liu, ”Automatic disfluency identification in conversational speech using multiple knowledge sources”, *Proc. Eurospeech*, pp. 957–960, 2003.

L. Mangu and E. Brill, ”Automatic rule acquisition for spelling correction”, *Proc. Intl. Conf on Machine Learning*, pp. 187–194, 1997.

L. Mangu and M. Padmanabhan, ”Error corrective mechanisms for speech recognition”, *Proc. ICASSP*, pp. 29–32, 2001.

M. Meteer et al., ”Dysfluency annotation stylebook for the Switchboard corpus”, Distributed by the LDC, 1995.

C. Nakatani and J. Hirschberg ”A corpus-based study of repair cues in spontaneous speech”, *Journal of the Acoustical Society of America*, pp. 1603–1616, 1994.

G. Ngai and R. Florian ”Transformation-based learning in the fast lane”, *Proc. NAACL*, pp. 40–47, 2001.

NIST, “The Rich Transcription Fall 2003 (RT-03F) evaluation plan,” `http://www.nist.gov/speech/tests/rt/rt2003/fall/docs/rt03-fall-eval-plan-v9.pdf`, 2003.

A. Ratnaparkhi, “A maximum entropy part-of-speech tagger”, *Proc. Empirical Methods in Natural Language Processing Conf.*, pp. 133–141, 1996.

E. Shriberg, *Preliminaries to a theory of speech disfluencies*, PhD thesis, Department of Psychology, University of California, Berkeley, 1994.

E. Shriberg et al., “A prosody-only decision-tree model for disfluency detection”, *Proc. Eurospeech*, pp. 2383–2386, 1997.

E. Shriberg et al., ”Prosody-based automatic segmentation of speech into sentences and topics” *Speech Communication*, 32(1-2), pp. 127–154, 2000.

K. Sönmez et al., “Modeling dynamic prosodic variation for speaker verification,” *Proc. Intl. Conf. on Spoken Language Processing*, v. 7, pp. 3189–3192, 1998.

A. Srivastava and F. Kubala ”Sentence boundary detection in Arabic speech”, *Proc. Eurospeech*, pp. 949–952, 2003.

A. Stolcke and E. Shriberg ”Automatic linguistic segmentation of conversational speech”, *Proc. ICSLP*, v. 2, pp. 1005–1008, 1996.

A. Stolcke et al., “Automatic detection of sentence boundaries and disfluencies based on recognized words,” *Proc. ICSLP*, 1998, v. 5, pp. 2247–2250.

A. Stolcke, ”SRILM - an extensible language modeling toolkit”, *Proc. ICSLP*, v. 2, pp. 901-904, 2002.

S. Strassel, “Simple metadata annotation specification version 5.0”, `http://www.ldc.upenn.edu/Projects/MDE/Guidelines/SimpleMDE\_V5.0.pdf`, 2003.